Tree Models

1. Decision Tree
   - Regression tree / Classification tree
   - Recursive binary split (greedy approach)
   - Pruning (Cost complexity pruning) → use cv to select alpha
   - Advantages and disadvantages
2. Bagging
3. Random Forest
4. Boosting

Decision Tree, bagging, random forests, and boosting

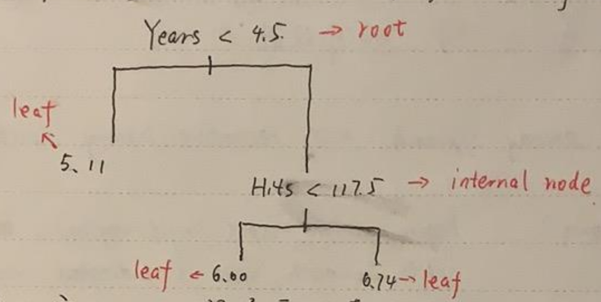
Decision tree 采用树形结构，层层推理来实现最终分类.

包含 {
- 根节点 (root)
- 内部节点 (internal node)
- 叶节点 (leaf / terminal node)

由上至下 (upside down)，通过对对应特征属性测试，分析决策结果

☆ Regression Tree

拿 ISLR 中的 Hitters 举例 (regression on a baseball player's Salary based on Years and Hits)



这里，Years 是决定 Salary 的最重要因素 (作为 root)，假定球员生涯时长小于 4.5 年 (less experienced)，那么击中数 (Hits) 显然对 Salary 影响不大，反之，影响很大

优点：1) easier to interpret 2) nice graphical representation

所以如何建立一个决策树呢？

(书上原文)

Step 1:

We divide the predictor space - that is, the set of possible values for $X_1, X_2, \cdots, X_p$ - into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.

Step 2:

For every observation, that falls into the Region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

假如，我们把 feature 分成若干（这里是两个）区所的域 (R_1, R_2)
the response mean of R_1, R_2 is 10, 20. 对 $\forall x \in R_1$, we predict 10.
反之亦然.

但 $R_1, \cdots, R_j$ 怎么建立呢？

The goal is to find boxes $R_1, \cdots, R_j$ that minimize the RSS

$$\Rightarrow \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

$\hat{y}_{R_j}$ 是 R_j 里所有 observation 对应的 response 的均值.

但是，这在计算上不可行，如何改进？

考点，敲黑板

☆ top-down, greedy approach $\Rightarrow$ recursive binary splitting

什么是 top-down? begin at the top (single region), ~~than~~ then successively splits; each split is indicated via two new branches further down on the tree.

为什么 greedy? because at each step of the process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in the future step.
（也就是 split rule 只根据当前情况确定，而不是在接下来的steps 再选更好的分割方式）

How to perform recursive binary splitting?

1) 取 X_j $\Rightarrow$ 2) set cutpoint s s.t.

$$X_j \to \{X \mid X_j < s\}, \quad \{X \mid X_j \geq s\}$$

（目的是 minimise RSS）

$\Downarrow$

3) repeat and find the best predictor and best cutpoint
也就是选最好的 j 和 s.

$\Leftarrow$ 4) Split one of the two previously identified regions

$\Leftarrow$ 5) Stop until a stopping criterion is reached

但是这个过程很可能 overfit，因为最终的结果太复杂。
原因：1) 错误的样本数据 2) 特征不能作为完全分类标准 3) 巧合性
So, a smaller tree with fewer splits (fewer regions) might lead to lower variance. (解决方法：比如把 minimize RSS 改成，decrease RSS 达到一定程度).
But, a split that leads to a large reduction in RSS later on.

所以，最好的做法是剪枝
grow a very large tree T_0, and then prune (修剪) it back to obtain a subtree.

We select a subtree that leads to the lowest test error rate. (Cross validation)

但是这个做计算量会很大，因为子集数太多了。咋办？

Cost complexity pruning (weakest link pruning)

目的：Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter α.
(α gives a measure of the value of sub-tree. i.e the reduction in error per leaf)
这里 α 也是一个 regularization term. 和 ~~reg~~ ridge / LASSO regression 和 SVM 里的 C 和 Gamma 一样用 CV 来选.

$$\alpha = \frac{R(t) - R(T_t)}{|f(T_t)| - 1}$$

where $R(T_t)$ — training error of a subtree $T_t \Rightarrow$ a tree with root at node t
$R(t)$ — training error of node t
$|f(T_t)| - 1$ is the number of leaves to prune.

下

For each value of α. there corresponds a subtree $T \subset T_0$ such that

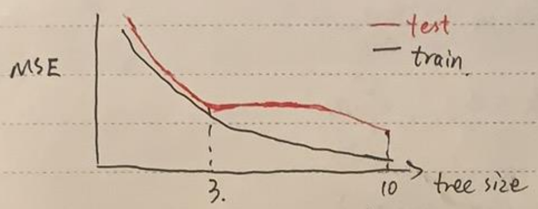
$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T.

$\alpha \uparrow$，修剪程度$\uparrow$，复杂程度$\downarrow$ ~~node~~ leaves $\uparrow$

如何选α → k-fold cross validation.

test error is function of α.



3-node tree is best even though it takes on its lowest value at 10-node tree.

☆ Classification Tree

Classification tree 和 regression tree 很像，但RSS不能用作二分裂的标准了。用 classification error rate / Gini index / cross entropy

复习: ① Classification error : $E = 1 - \max_k (\hat{p}_{mk})$

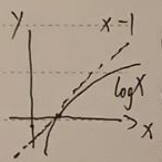
($\hat{p}_{mk}$ => proportion of training observations in the mth region that are from the kth class, 也就是 $P(Y=k|X)$ 或 $h_\theta(x)$)

② Gini index (impurity) : $G = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$

越小越纯 ($\hat{p}_{mk} \approx 1$ or 0)

③ Cross entropy : $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

(把 $\log \hat{p}_{mk}$ 替换成 $-(1-\hat{p}_{mk})$ 就是 Gini index)

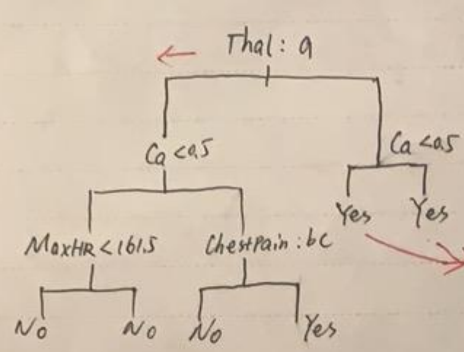


所以用它们哪个？ 如果追求 accuracy，在修剪时用 E.

还有 decision trees 不光可分 continuous values, qualitative 也可以

和 regression tree 一样，先建立原始的 Tree，然后 prune，用 CV 调整 nodes 数量。

例：



思考：
两个 Yes 为什么要分???

优： ① easy to explain (even easier than linear regression)
② more closely mirror human decision making
③. can be displayed graphically
④. easily handle qualitative predictors without the need to create dummy variables

缺： ①. accuracy not that good
②. a small change in the data can cause a large change in the final estimated tree.

Bagging

还记得 bootstrap 吗?

It's used in many situations in which it's hard or even impossible to directly compute the standard deviation of a quantity of interest.

bagging (Bootstrap aggregation) is a procedure for reducing the variance of a statistical learning method (又是 overfitting)

简单说，计算

$\hat{f}^1(x), \hat{f}^2(x), ---, \hat{f}^B(x)$ using B separate training set, and average them to obtain a low-variance model.

given by

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{b}(x)$$

Bootstrap 帮我们从一个 training set 里有放回的重复抽样，代替了建立多个 training set.

$$\Rightarrow \hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

We train our method on the bth bootstrapped training set to get $\hat{f}^{*b}(x)$, and average.

For improving regression trees:

We construct B regression trees using B bootstrapped training sets, and average the resulting predictions.

Bagging Improves the accuracy by combining together hundreds of even thousands of trees into a single procedure.

How about classification tree?

just like KNN (voting algorithm), we have records from the predictions by each of the B trees. The overall prediction is the most commonly occuring class among the B predictions.

注意：B 就算过大，不会 overfitting, 只是计算时间长

How to evaluate bagging models?

这里不用 CV 来 estimate test error.

介绍 OOB (out of bag)

相当于每个 bagging tree 都假设用了 $\frac{2}{3}$ 的数据作为 training set，剩下 $\frac{1}{3}$ 用来 predict，那么每一个 observation (或者 ith observation) 在一个 tree 时有一个 prediction，B 个 tree 有 $\frac{B}{3}$ 个 prediction，之后我们取平均值 (regression) 或众数 (classification)，这就有了一个 OOB prediction (对于 ith observation)，那么对每个 observation 都这么做，得到 OOB MSE (~~对~~ regression) 或 classification error.

所以 OOB error is a valid estimate of the test error for the bagged model. 所以用 OOB error 在面对大型的 dataset 更好.

注意: bagging improves the accuracy at the expense of interpretability.

但对于 variable importance, 我们可以用 RSS 或 Gini index 来得到 overall summary of the importance of each predictor.
对于 bagging regression tree, 我们记录在一个选定的 predictor 分裂时, RSS 下降的总量, 对 B 个 tree 的情况取平均 (classification 同理)

Random Forests.

那么 bagged trees 还能再提升么? YES!

Random forests provide an improvement over bagged trees by way of a small tweak the decorrelates the trees.

什么是 decorrelate? 为什么 decorrelate?
相当于在每次分裂时, 我们不考虑所有的 predictors, 只取它们的一部分
假如, dataset 里有一个 strong predictor, 一些 moderately strong predictors.
那么大多数的 tree 都用 strong preditor 作为 root (top 的 split).
这样下来, 几乎所有的 tree 都差不多.
(The predictions from the bagged trees will be highly correlated).
对这些 highly correlated 结果取均值没意义, 是无法降低 variance 的.

所以, 只取 ~~的~~ 所有 predictors 的子集, $m=\sqrt{p}$ (一般来说)
平均 $(p-m)/p$ 个分裂过程不会考虑 strong predictor, 所以没那么强的 predictor 会更多被使用。 model → less variable → more reliable

Random Forest 与 bagging 区别: m 的选取
bagging : $m=p$.
RF : $m=\sqrt{p}$ (default)

如果我们有许多 correlated predictors，m 取小一点。
这里 B 很大也不会 overfitting. 所以 B 要取足够大。

Boosting （书上说的是 Gradient Boosting）
Boosting 从另一个角度改进了 decision tree，同样沿用了 bagging 的思想（把 training set bootstrap 成多个 training set，得到多个 tree，最后求平均）. 但 Bagging 中的 tree 都是 independent.

而 Boosting 是使每个 tree 的建立都基于之前的 tree
(trees are growing sequentially)
所以 Boosting 没用 bootstrap sampling.

步骤：
① Train a decision tree
② Apply the tree to predict
③ Calculate the residual of this tree. Save residual errors as new y
④ repeat step 1 until we reach B.
⑤ final prediction

Key: Boosting learns slowly, learns from the mistake (residual error)

Boosting has 3 tuning parameters.
1. B (the number of trees) : B is too large, overfit
CV to select B
2. λ (shrinkage parameter) : a small positive number.
This controls the learning rate (0.01, 0.001)
λ too small $\longrightarrow$ B large to have good performance
3. d (the number of splits in each tree) : This ~~contro~~ controls the complexity of the boosted ensemble. Often d=1 works well $\Rightarrow$ each tree is a stump (top split)
树桩